

Распределённая обработка платежей с помощью Apache Ignite

Николай Кувыркин



Николай Кувыркин

Senior Architect @ Raiffeisen



- Более 20 лет в разработке
- С начала 2000-х в бэкенде, основной опыт связан с базами данных.
- Биллинг для первых WiFi хот-спотов в Москве (отели Mariott)
- 10 лет работал в Лаборатории Касперского
  - распределённая файловая система KLDFS
  - облачная система детектирования файлов WhiteListing
- Последние 3 года работаю в Compliance Райффайзенбанк
  - KYC — система автоматического обновления данных клиентов банка
  - StarkNG — система обработки потока платежей



История: зачем нужен этот проект

- Stark — CRM для офицеров Compliance
- Большой поток платежей → необходимость автоматических решений
- Большая нагрузка, высокие требования к скорости обработки
- Сложная логика правил → ML
- Автоматические решения → новая система



Требования

- Высокая степень параллелизма
- Стабильное время ответа в широком диапазоне нагрузок
- Максимальная “наблюдаемость” системы
- Лёгкая и быстрая модификация бизнес-логики
- Возможность ретро-тестирования
- Обновление всех частей системы без простоя
- Масштабирование



Обработка платежа

Получение данных

Обогащение данных

Обработка моделью

Ответ



Обработка платежа

Платформа

Получение данных

Обогащение данных

Ответ

Модель

Обработка моделью



Архитектура

Stark

Datalake

Платформа

Модель

MQ Brokers



Схема развертывания

Серверы vRA

Node ... Node

Client nodes

Node1 ... NodeN

Внешние системы

Kubernetes

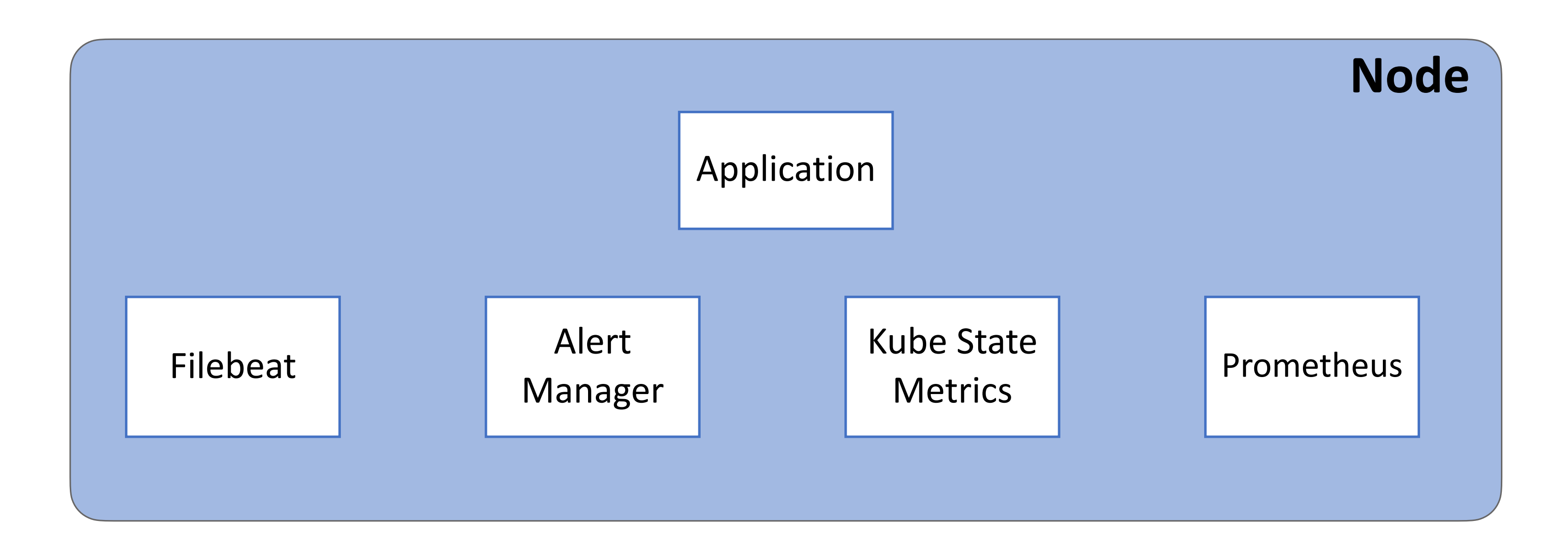
Балансировщик

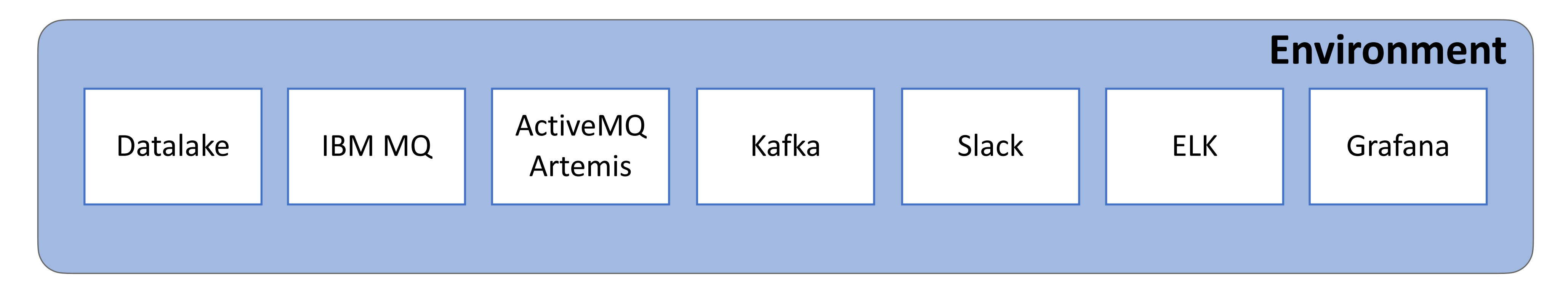
Model nodes

Node1 ... NodeN



Client Node


Node
Application
Filebeat
Alert Manager
Kube State Metrics
Prometheus


Environment
Datalake
IBM MQ
ActiveMQ Artemis
Kafka
Slack
ELK
Grafana


HL
HighLoad++
2022

Производительность

Среда	Количество запросов	Среднее время ответа (ms)
TKGI	240 519	569
vRA	238 676	81

- TKGI не подходит из-за медленного балансировщика (NSX)
- На собственном кластере 👌



Ресурсы и отказоустойчивость

- Серверные ноды Платформы:
  - Минимум 2 экземпляра — на данный момент 5 x (8 CPU, 32Gb RAM, 250Gb SSD)
  - Хранят данные, данные на нодах различаются
  - Степень избыточности данных можно настраивать
  - Отказоустойчивость обеспечивается Apache Ignite



Ресурсы и отказоустойчивость

- Кластер K8S для Платформы и Моделей:
  - Master Node: 1 x (8 CPU, 10Gb RAM, 50Gb Disk)
  - Worker Node: 4 x (12 CPU, 16Gb RAM, 150Gb Disk)



Ресурсы и отказоустойчивость

- Клиентские поды Платформы и Моделей:
  - Минимум 1 экземпляр (масштабирование средствами K8S)
  - Не хранят состояние
  - Идентичные
  - Быстрый старт новых подов средствами K8S
  - Отказоустойчивость обеспечивается K8S



Метрики Apache Ignite

- Apache Ignite: jmx-prometheus exporter
  - Cluster metrics
    - Total cache size
    - Rebalance partition count
    - Cluster owning partition count
    - Cluster moving partition count



Метрики Apache Ignite

- Apache Ignite: jmx-prometheus exporter
  - Node metrics
    - Total baseline nodes
    - Active baseline nodes
    - Node owning partition count
    - Partition count
    - Cache size
    - Node moving partition count



Метрики Apache Ignite

- Apache Ignite: jmx-prometheus exporter
  - Cache metrics:
    - Valid for reading
    - Valid for writing



Технологии

- Платформа:
  - Apache Ignite
  - Java (8 → 11 → 17)
  - Spring
  - IBM MQ
  - Apache ActiveMQ Artemis
  - Apache Kafka
  - Apache Hive
- Модель:
  - MRMtool → Py4J → Flask
  - Python
- ML-правила
  - Datalake
  - Python, Spark, PySpark, HiveQL, Apache Airflow, NiFi



Конвейеры кода

- Все:
  - Типовой CI/CD (GitLabCI)
  - Развертывание средствами платформы без простоя
  - Интеграционные тесты
  - Версионирование
- ML-правила и витрины:
  - Ревью командой Datalake



Выпуск проекта в промышленную эксплуатацию

Входная очередь

Legacy system

Выходная очередь

New system



Выпуск проекта в промышленную эксплуатацию

Входная очередь

Legacy system

Выходная очередь

New system

Тестовая выходная очередь



Выпуск проекта в промышленную эксплуатацию

Входная очередь

Legacy system

Выходная очередь

New system



Проблемы

- Массовые платежи от одного клиента
- Split Brain кластера Apache Ignite



Массовые платежи от одного клиента

Клиенты обрабатываются **параллельно**

Client A → Поток 1

Client B → Поток 2

Client C → Поток 3



Массовые платежи от одного клиента

Клиенты обрабатываются **последовательно**

Клиент	Поток
Client A	Поток 1
Client A	Поток 2
Client A	Поток 3

Поток 2 ждёт поток 1

Поток 3 ждёт поток 2



Массовые платежи от одного клиента

Клиенты обрабатываются **параллельно**

Клиент	Поток
Client A	Поток 1
Client B	Поток 2
Client C	Поток 3

Поток 1 → Client A → Deferred Queue (Client A)

- ClientInFlight Counter
- LockTime



Split Brain в кластере Apache Ignite

Node 1
MP1
BP3
BP4
MP5

Node 2
BP1
MP2
BP4
BP5
MP6

Node 3
BP1
BP2
MP3
BP5
BP6

Node 4
BP2
BP3
MP4
BP6







Split Brain в кластере Apache Ignite

Node 1	Node 2	Node 3	Node 4
MP1	BP1	BP1	
	MP2	BP2	BP2
BP3		MP3	BP3
BP4	BP4		MP4
MP5	BP5	BP5	
	MP6	BP6	BP6



Master Partition



Backup Partition



Split Brain в кластере Apache Ignite

Node 1
MP1
BP4
BP5
MP6

Node 2
BP1
MP2
BP5
BP6

Node 3
BP1
BP2
MP3
BP6

Node 4
BP2
BP3
MP4

Node 5
BP3
BP4
MP5







Split Brain в кластере Apache Ignite

Node 1
MP1
BP4
BP5
MP6

Node 2
BP1
MP2
BP5
BP6

Node 3
BP1
BP2
MP3
BP6

Node 4
BP2
BP3
MP4

Node 5
BP3
BP4
MP5



Master Partition



Backup Partition



Split Brain в кластере Apache Ignite

Node 1
MP1
BP4
BP5
MP6

Node 2
BP1
MP2
BP5
BP6

Node 3
BP1
BP2
MP3
BP6

Node 4
BP2
BP3
MP4

Node 5
BP3
BP4
MP5







Вероятность Split Brain в кластере Apache Ignite с 2 Backup Partitions

2 nodes leave the cluster

- 4 nodes cluster: 100%
- 5 nodes cluster: 5/10 = 50%
- 6 nodes cluster: 3/15 = 20%
- 7+ nodes cluster: 0%



Фишка: удаление данных с помощью ScanQuery

```
private static class MyFilter implements IgniteBiPredicate<BinaryObject, BinaryObject> {


        public boolean apply(BinaryObject key, BinaryObject value) {
            if (value.<Integer>field("field") > 100)
                return true;
            return false;
        }
    }

...

    ignite.cache("MyCache").query(new ScanQuery<>(new MyFilter()));
```



Фишка: удаление данных с помощью ScanQuery

```
private static class MyFilter implements IgniteBiPredicate<BinaryObject, BinaryObject> {
        @IgniteInstanceResource
        private transient Ignite ignite;

        public boolean apply(BinaryObject key, BinaryObject value) {
            if (value.<Integer>field("field") > 100)
                ignite.cache("MyCache").withKeepBinary().remove(key);
            return false;
        }
    }

...

    ignite.cache("MyCache").query(new ScanQuery<>(new MyFilter()));
```



Резюме

- Стандарты, стандарты, стандарты
- Правильный выбор технологий
- Использование принятых в организации практик разработки
- Тестирование применимости в конкретном проекте имеющихся в организации решений
- Тестирование в режиме параллельной работы с существующей системой



Обратная связь, комментарии и вопросы по докладу:

- Telegram: @R2AMR
- Email: nkuvyrkin@mail.ru

